**Programación I**
Curso 13/14

| |
|---|
| Apellidos, Nombre: |
| Grupo: |
| Num. Matricula: |

# Eval2

17/01/2013

# SOLUCIÓN

### Ejercicio 1

En el juego del dominó las fichas que se utilizan están divididas en dos cuadrados, el izquierdo y el derecho, cada uno de los cuales lleva marcados de uno a seis puntos, o no lleva ninguno. Define un tipo tupla[1] *Ficha* que represente una ficha del dominó. **(0,5 puntos)**

---

```
//Una ficha del dominó.
class Ficha {
  int iz; //Valor izquierdo de la ficha;
  int de; //Valor derecho de la ficha;
  Ficha () { }
  Ficha (int valorIz, int valorDe) {
    iz = valorIz;
    de = valorDe;
  }
}
```

---

Define datos como variables de tipo *Ficha* que representen las siguientes fichas: *blancaDoble* (0 puntos en los dos cuadrados), una ficha *blancaPito* con 0 puntos en el cuadrado izquierdo y 1 en el derecho y una ficha *cincoSeis* con 5 puntos en el cuadrado izquierdo y 6 en el derecho. **(0,5 puntos)**

---

```
Ficha blancaDoble = new Ficha(0, 0);
Ficha blancaPito = new Ficha(0, 1);
Ficha cincoSeis = new Ficha(5, 6);
```

---

Dado el siguiente programa

```
class Ejercicio
{
  int change (int m, int n)
  {
    m = (m + 10) % 7 + 1;
    n = (n + 10) % 7;
    return n;
  }
```

[1] También conocido como registro u objeto.

```
  Ficha whose (int n, Ficha f)
  {
    n = n + 10;
    f.iz = change(f.iz, f.de);
    return f;
  }
  String watch (Ficha f)
  {
    return "(" + f.iz + ", " + f.de + ")";
  }
  Ficha f = new Ficha(5, 6);
  int n = 7;
  String prueba1 = watch(whose(n, f));
}
```

Indica: **(a)** el resultado de la variable *prueba1*; **(b)** el contenido de la variable *n* tras ejecutar *prueba1* y **(c)** de la variable *f* tras ejecutar *prueba1*. **(1 punto)**

---

```
(a) (2, 6)
(b) 7
(c) (2, 6)
```

---

Define un array que contenga las tres fichas de dominó que definiste previamente en el orden dado y otro que esté vacío. **(1 punto)**

---

```
Ficha[] serie3 = {blancaDoble, blancaPito, cincoSeis};
Ficha[] vacio = {};
```

---

Define un visualizador para el array de fichas. **(1 punto)**

---

```
  String aString (Ficha[] a)
  {
    String s = "{ ";
    for (int i=0; i<=a.length-2; i++)
      s = s + watch(a[i]) + ", ";
    if (a.length!=0)
      s = s + watch(a[a.length-1]);
    return s + " }";
  }
```

---

**Ejercicio 2**

En el juego del Dominó dos fichas sólo pueden colocarse juntas cuando los cuadrados adyacentes tengan los mismos puntos. Se dice que una serie de fichas es correcta si todas sus fichas están colocadas siguiendo esta regla.
Suponiendo que un array representa la serie de fichas que están encima de la mesa en un momento dado, definir una función *serieCorrecta* que devuelva cierto si y solo si la serie de fichas que recibe como parámetro de entrada es correcta y falso en caso contrario.
Dada la ficha *seisBlanca* (con 6 puntos en el cuadrado izquierdo y 0 en el derecho) indica qué devolverá la función *serieCorrecta* si recibe como arrays de fichas:

1. El array vacio.
2. El array formado por las fichas *blancaDoble*, *blancaPito*, *cincoSeis* y *seisBlanca* en ese orden.
3. El array formado por las fichas *cincoSeis*, *seisBlanca*, *blancaDoble* y *blancaPito* en ese orden.

**(1 punto)**

---

1. `true`
2. `false`
3. `true`

---

Define la función *serieCorrecta* en Java. **(3 puntos)**

---

```
boolean serieCorrecta (Ficha[] serie)
{
  boolean esCorrecta = true;
  int i = 0;
  while (esCorrecta && i <= serie.length-2)
  {
    if (serie[i].de != serie[i+1].iz)
      esCorrecta = false;
    else
      i = i+1;
  }
  return esCorrecta;
}
```

---

**Ejercicio 3**

Dadas las siguientes funciones

```
boolean flop (char c)
{
  return 'A'<=c && c<='Z';
}
char[] fly (char[] a, int i, int j)
{
  char t = a[i];
  a[i] = a[j];
  a[j] = t;
  return a;
}
char[] flow (char[] a, int pos)
{
  for (int i=a.length-1; i>=pos+1; i--)
    if ( !(a[i-1] <= a[i]) && !flop(a[i-1]) && !flop(a[i]) )
      a = fly(a, i-1, i);
    return a;
}
char[] flaw (char[] a)
{
  for (int i=0; i<a.length; i++)
    a = flow(a, i);
  return a;
}
```

Indica cuál es el valor devuelto por *flaw* si recibiera como argumento tu nombre y tus apellidos (sin espacios en blanco entre ellos y con la letra inicial del nombre y de los apellidos en mayúscula y el resto en minúsculas). Tratar las letras ñ y Ñ como gn y GN. Es decir, un nombre como LuisRobleAñoso debe tratarse como LuisRobleAgnoso. **(1 punto)**

---

Para:

```
nombre = { 'L', 'u', 'i', 's',
           'R', 'o', 'b', 'l', 'e',
           'A', 'g', 'n', 'o', 's', 'o' }
```

Devuelve:

```
flaw (nombre) --> { 'L', 'i', 's', 'u',
                    'R', 'b', 'e', 'l', 'o',
                    'A', 'g', 'n', 'o', 'o', 's' }
```

---

Indica qué hace *flaw*: **(1 punto)**

---

Ordena de menor a mayor orden alfabético las letras minúsculas de cada palabra de *a*. Cada palabra viene separada por una letra mayúscula. Las letras mayúsculas se dejan donde están. *a* no puede contener el carácter 'ñ'.

---

**NOTAS:**

- El tiempo para realizar la prueba es de **1 hora y 30 minutos**.